Raytheon

High Performance Situation Display Capability for the CNS/ATM Domain

Jean-Marie Dautelle
Jean-Marie_R_Dautelle@raytheon.com

Waseem Naqvi
Waseem_Naqvi@raytheon.com

Raytheon
Network Centric Systems
Marlborough, MA, 01752
Tel: 508.490.3635

Raytheon

Outline

- Motivation
- State of the Art
- Architecture
- Real-time constraints
- Recording
- Summary

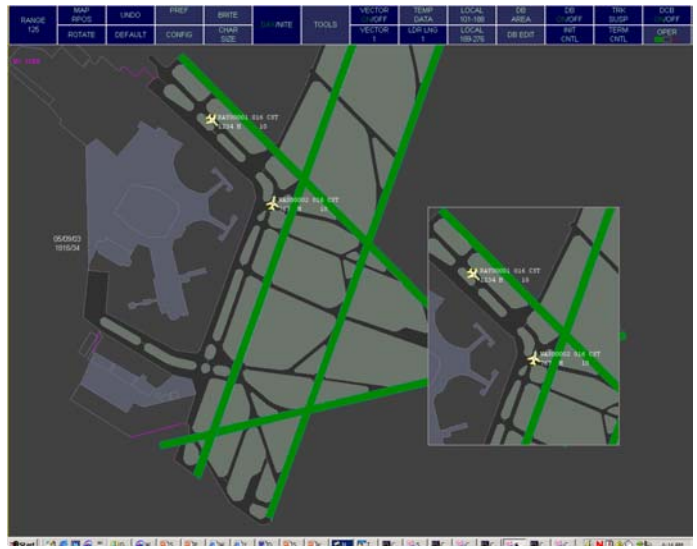
Raytheon has Deployed many CNS/ATM Systems Worldwide

Raytheon

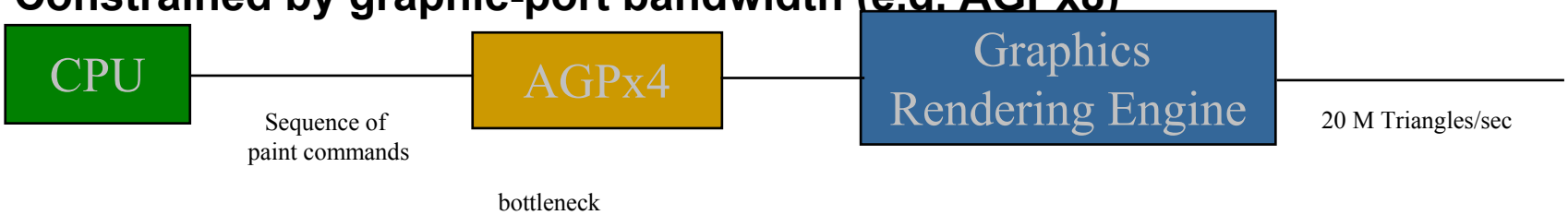
Customers have unique display preferences!

Current State-of-the-Art

- Displays are component based
  - Windows, Icons, Mouse, Pointers (WIMP)
    - ABC Keyboards, range dials, examples of ease of use



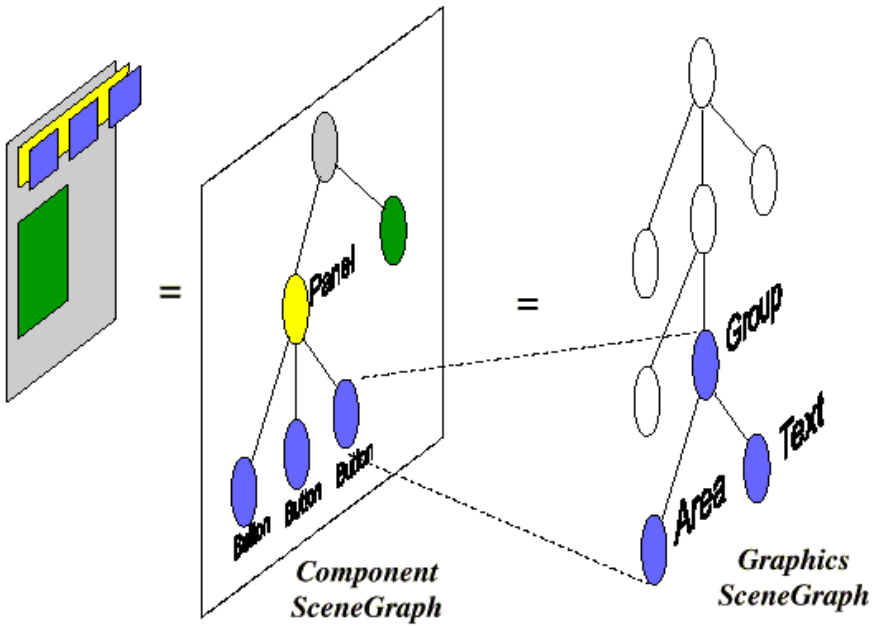
- Current display components are software rendered (sequence of paint commands)
  - CPU intensive
  - Constrained by graphic-port bandwidth (e.g. AGPx8)



The graphics hardware isn't used to advantage

Our Approach

- Current graphic toolkits decompose complex objects into sequence of paint operations, the hardware doesn't get the big picture
- New 3D graphics hardware are optimized to render virtual models (scene graph) rapidly – without CPU interaction



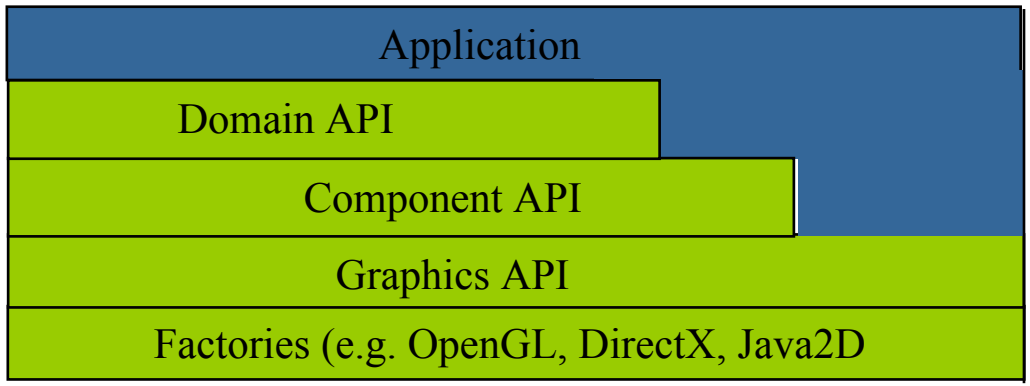
Our approach

- Upload our virtual model of the complete display to the hardware for extremely rapid rendering
- Performance is >10x faster than traditional method

Let the hardware render the display

Architecture & Implementation

- Layered Architecture



- Platform independence
  - Abstract Factory Pattern
  - Implementation using Java™
- Standard component interface
  - Swing-like (Java graphics standard)
    - Training/Tutorials
    - Documentation
    - Ready access to skills base

Best practices employed

Java and Real-Time Constraints

"Rendering of Situation Displays (e.g. Air Traffic Control) has to be performed on demand and has to be time-bounded"

- **Issues with Java**
  - **Java is non-deterministic**
    - **Use of incremental garbage collection (GC)**
    - **Scene-graph hardware based rendering is not affected by Java GC**
  - **Performance**
    - **Rendering is performed by the hardware. The main CPU is free to carry on others time-critical tasks (e.g. input/output)**

The display framework has addressed Java shortcomings

ATC Displays

In the field of Air Traffic Control, it is desirable to be able to record/playback the data being displayed to the controller in case of accident, for simulation purposes, training, etc.

Recording needs to be 100% accurate.

Conventional techniques include:

- **Hardware recording (e.g. video recorder) generating a huge amount of data.**
- **Software recording requiring additional code to record/playback accurately every possible action and state.**

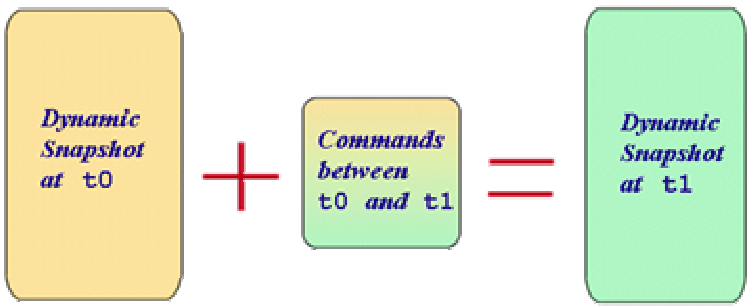
C2/ATMS systems require recording of events

Patented Dynamic Snapshot Approach

Raytheon

"Instead of recording the snapshot state itself, we record the minimal set of commands, which would put the system in the snapshot state"

This sequence of commands is referred as "The dynamic Snapshot".



During recording or playback, it is possible to "move" the dynamic snapshot through time without affecting the operation of the system in a substantive way.

Snapshot are transparently captured

Advantages of the Dynamic Snapshot Approach

Raytheon

- No need to serialize/de-serialize the objects (only the commands).
- The default object's state does not need to be recorded.
- If the commands are recorded (delta) and time-stamped, then it is possible to continuously playback the changes to the system from any "snapshot" point.
- Seeking a particular time, can be performed very quickly by moving the dynamic snapshot forward and then playing the "translated snapshot" commands (minimal set).

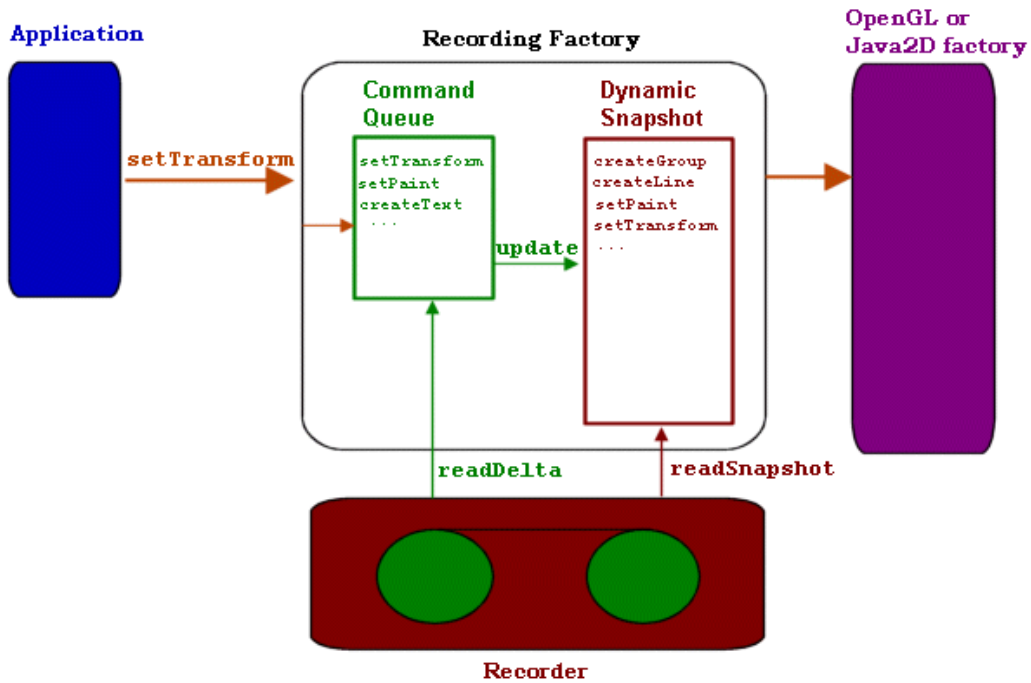
Lightweight approach provides significant advantages

Integration with our graphic toolkit

Raytheon

Integration was seamless and limited to the creation of a new graphic factory, within our graphics toolkit

"The recording factory", acted as a proxy and forwarded the commands to the real factory responsible for the rendering operations.



Dynamic snapshots and deltas are captured transparently

Performance

Raytheon

On a typical ATC display at maximum load (1000+ targets):

- Recording takes less than 2% of total CPU usage and is performed asynchronously by background tasks (no impact on rendering performance).
- The recording data rate is less than 56Kbits/second.
- The “seeking” time to internally playback up to 10 minutes of data (the snapshot period) is less than 1 second.
- The dynamic snapshot size “stabilizes” at about 100 Kbytes.

The Dynamic Snapshot approach has proved to be scaleable

Raytheon

Summary

- Very fast (takes full advantage of hardware acceleration)
- Allows real-time display applications to be written using Java (higher productivity than C++)
- Small footprint
- Uses standard architecture: Short learning curve for Java programmers familiar with standard “Swing”
- Transparent record/playback capability
- Entirely customizable using XML (component layout, keystroke binding, button’s action mapping, etc...)
- Runs on any platform with a Java Virtual Machine or Java Compiler (e.g. GNU Java Compiler)

Our framework allows the building of CNS/ATM displays rapidly


RANGE 125
MAP RPOS
ROTATE
UNDO
DEFAULT
PREF
CONFIG
BRITE
CHAR SIZE
DAY/NITE
TOOLS
VECTOR ON/OFF
VECTOR 1
TEMP DATA
LDR LNG 1
LOCAL 101-188
LOCAL 189-276
DB AREA
DB EDIT
DB ON/OFF
INIT CNTL
TRK SUSP
TERM CNTL
DCB ON/OFF
OPER
NO SLEW
RAY00001 016 CST
1234 H 10
WAS00002 016 CST
767 H 10
05/09/03
1816/34
RAY00001 016 CST
1234 H 10
WAS00002 016 CST
767 H 10